How-To: Query ElasticSearch

This process is used to query simple counts from data in ElasticSearch within EDH. These queries could be used to understand the data within a client's database before and after a monthly data load. We must understand how a data load alters a client's data and these counts help do just that.

Step-by-step guide

This guide is based on the work in PRED-4293 and focuses on the use of Sense as the UI to query the data.

Check out this primer on Sense and UI Guide if this is your first time using the tools. Between these guides and the examples below, you can build your own queries to explore the data in ElasticSearch.

Other useful links for using Sense:

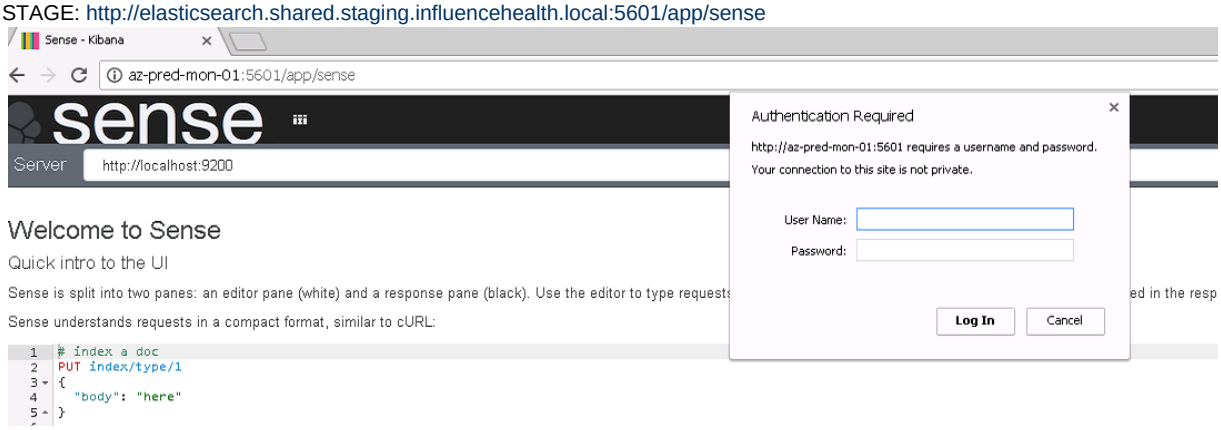
- Sense - a Cool JSON Aware Interface to Elasticsearch
- ElasticSearch 101 – a getting started tutorial
- All About the Sense Web Application

This query process accomplishes the following:

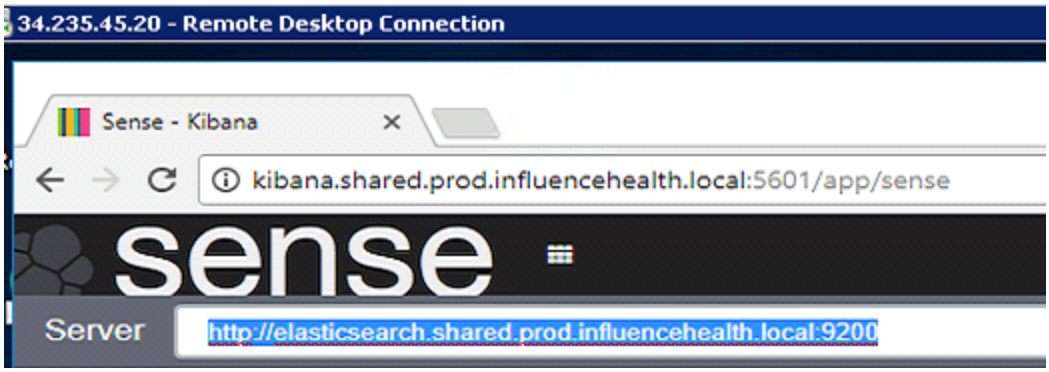
1. Displays current data snapshot of current data with data points such as:
    a. Count of total persons - to understand total population within database
    b. Count of total households - to understand grouping of families within database

Query Process Steps

1. RDP to AI Bastion Server for the environment needed.
    a. IP Addresses
        i. PROD: 34.235.45.20
        ii. STAGE: 34.202.138.26
    b. Login Creds
        i. Local to machine, this is an AWS Windows box. Help Ticket is needed if you do not have a login. It will be first.last but password is for native login, not AD creds.
2. Launch browser and navigate to
    a. PROD: http://kibana.shared.prod.influencehealth.local:5601/app/sense
    b. STAGE: http://elasticsearch.shared.staging.influencehealth.local:5601/app/sense

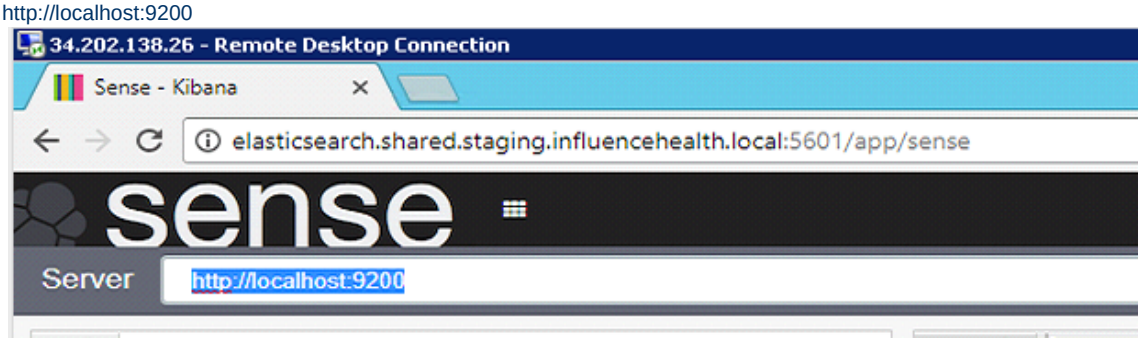


3. Login using;
    a. Username: azureuser
    b. Password: See Nathan Cooke, Wesley Brown, or Kirk Alford if you need the password.
4. Update the server box on this page to point one of the nodes in the cluster
    a. PROD
        i. http://elasticsearch.shared.prod.influencehealth.local:9200
        ii.



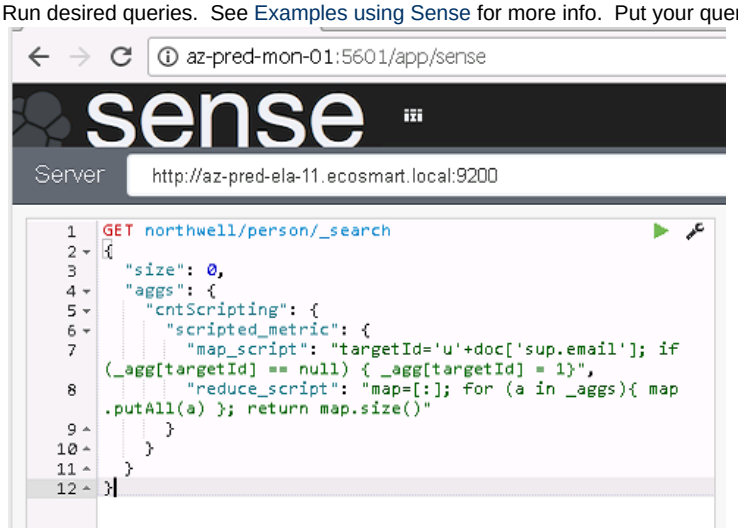
b. Stage

i. http://localhost:9200

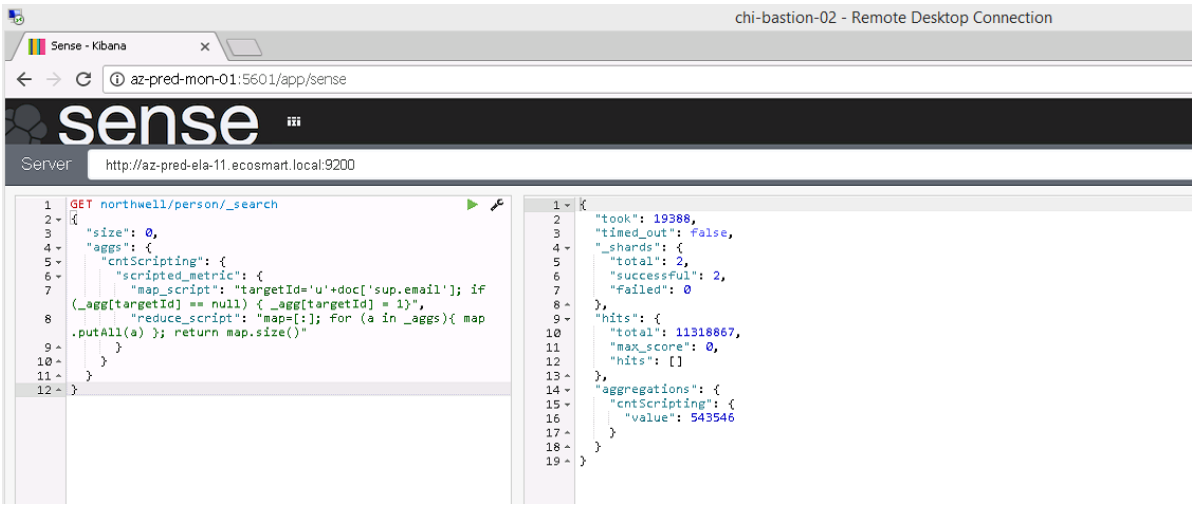


ii.

5. Run desired queries. See Examples using Sense for more info. Put your query in the window on the left.



6. Click the green triangle to execute command.
7. There is really no indication that the query is running, the results will be displayed in the right window.



Examples using Sense

Count of Distinct emails

```
GET <clientKey>/person/_search
{
  "size": 0,
  "aggs": {
    "cntScripting": {
      "scripted_metric": {
        "map_script": "targetId='u'+doc['sup.email']; if (_agg[targetId] == null) { _agg[targetId] = 1}",
        "reduce_script": "map=[:]; for (a in _aggs){ map.putAll(a) }; return map.size()"
      }
    }
  }
}
```

Total number of persons in a client's index

```
GET <clientKey>/person/_count
```

List of indexes in the database

```
GET _cat/aliases
```

Household Count

```
GET <clientKey>/person/_search
{
    "size": 0,
    "aggs": {
        "households": {
            "scripted_metric": {
                "init_script": "_agg = [:]",
                "map_script": "if (!_agg[doc.addressId.value] ) {
_agg.put(doc.addressId.value.toString(), null)}",
                "reduce_script": "count = 0;for (a in _aggs) {  for
(doc in a) {  count++;  }};return count;"
            }
        }
    }
}
```

The 30 most used email addresses in the client's dataset

```
GET <clientKey>/person/_search
{
  "size": 0,
  "aggs": {
    "email_count": {
      "terms": {
        "field": "sup.email",
        "size": 30
      }
    }
  },
  "query": {
    "bool": {
      "filter": {
        "bool": {
          "must": {
            "exists" : { "field": "sup.email"}
          }
        }
      }
    }
  }
}
```

Count of persons on DNS

```
GET <clientKey>/person/_search
{
  "size": 0,
  "aggs": {
    "pType": {
      "terms": {
        "field": "dns",
        "size": 10
      }
    }
  }
}
```

Person count by payer group for a specific mxGroup

```
GET <clientKey>/person/_search
{
  "size": 0,
  "query": {
    "nested": {
      "path": "encounters",
      "query": {
        "term": {
          "encounters.mxGroups": {
            "value": "12300"
          }
        }
      }
    }
  },
  "aggs": {
    "payertype": {
      "terms": {
        "field": "payerType",
        "size": 100
      }
    }
  }
}
```

All fields for 10 people

```
GET <clientKey>/person/_search
```

Propensity and Scores for persons

```
GET <clientKey>/person/_search
{
  "_source": ["propensities.id", "propensities.score"]
}
```

Count of persons with each propensity

```
GET <clientKey>/person/_search
{
  "size": 0,
  "aggs": {
    "props": {
      "nested": {
        "path": "propensities"
      },
      "aggs": {
        "prop": {
          "terms": {
            "field": "propensities.id",
            "size": 40
          }
        }
      }
    }
  }
}
```

Count of persons for each Recipe

```
GET <clientKey>/person/_search
{
  "size": 0,
  "aggs": {
    "recipes": {
      "nested": {
        "path": "recipes"
      },
      "aggs": {
        "recipe": {
          "terms": {
            "field": "recipes.id",
            "size": 50
          }
        }
      }
    }
  },
  "_source": ["recipes.id", "recipes.score"]
}
```

Count of persons by person type

```
GET <clientKey>/person/_search
{
  "size": 0,
  "aggs": {
    "pType": {
      "terms": {
        "field": "personType",
        "size": 10
      }
    }
  }
}
```

Related articles

- Postgres Buildout
- SG2 Groupings
- De-Identified Data Scope Document
- List Migration
- How-To: Query ElasticSearch